Workload Identity in Multi System Environments
Internet-Draft
Intended status: Standards Track
Expires: 27 March 2026

A. Schwenkschuster
SPIRL

23 September 2025

WIMSE Workload-to-Workload with HTTP Message Signatures
draft-schwenkschuster-s2s-http-sig-00

Abstract

This document defines an HTTP Message Signatures-based profile for workload-to-workload authentication within the WIMSE (Workload Identity in Multi System Environments) architecture. This profile uses the Workload Identity Token (WIT) combined with HTTP Message Signatures to provide cryptographic proof of possession and message integrity protection. The profile leverages RFC 9421 to sign HTTP requests and optionally responses, ensuring that workloads can authenticate each other and verify message integrity at the application level, particularly in environments where end-to-end TLS is not feasible due to middleboxes or other infrastructure constraints.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://ietf-wg-wimse.github.io/draft-ietf-wimse-s2s-protocol/draft-ietf-wimse-s2s-protocol.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-schwenkschuster-s2s-http-sig/.

Discussion of this document takes place on the Workload Identity in Multi System Environments Working Group mailing list (mailto:wimse@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/wimse/. Subscribe at https://www.ietf.org/mailman/listinfo/wimse/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-wimse/draft-ietf-wimse-s2s-protocol.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Copyright Notice

Table of Contents

1. Introduction

This document specifies an HTTP Message Signatures-based authentication profile for workload-to-workload communication as part of the WIMSE architecture defined in [I-D.ietf-wimse-arch]. This profile provides a standardized approach for workloads to authenticate each other and protect message integrity at the application level.

In many modern deployment environments, particularly in containerized and cloud-native architectures, workloads cannot rely on end-to-end TLS for security due to the presence of load balancers, API gateways, service meshes, and other intermediary systems. These middleboxes often terminate TLS connections, requiring application-level protection mechanisms to ensure workload authentication and message integrity.

This profile builds upon the base WIMSE workload-to-workload protocol and specifies the use of HTTP Message Signatures [RFC9421] as the proof of possession mechanism. The approach combines:

1. The Workload Identity Token (WIT) - a JWT that establishes the workload’s identity and binds it to a cryptographic key

2. HTTP Message Signatures - providing cryptographic proof that the sender possesses the private key corresponding to the public key in the WIT

3. Message integrity protection - ensuring that critical parts of HTTP requests and responses have not been modified

This profile is particularly suitable for deployments that require strong message integrity guarantees and the ability to sign both requests and responses. It provides protection against message tampering by intermediaries while allowing for the flexibility needed in complex multi-tier architectures.

The profile defines specific requirements for which HTTP message components must be signed, signature parameters that must be included, and validation procedures that recipients must follow. It is designed to be interoperable with other WIMSE authentication profiles, allowing different workload pairs within the same call chain to use different authentication methods as appropriate for their deployment environment.

2. Conventions and Definitions

All terminology in this document follows [I-D.ietf-wimse-arch].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Authentication Based on HTTP Message Signatures

This option uses the Workload Identity Token (WIT) as defined in Section 3.1 "The Workload Identity Token" of [I-D.ietf-wimse-s2s-protocol] and the private key associated with its public key, to sign the request and optionally, the response. See Section 4 "Security Considerations" of [I-D.ietf-wimse-s2s-protocol] for security considerations. This section defines a profile of the Message Signatures specification [RFC9421].

The request is signed as per [RFC9421]. The following derived components MUST be signed:

* @method
* @request-target

In addition, the following request headers MUST be signed when they exist:

* Content-Type
* Content-Digest
* Authorization
* Txn-Token [I-D.ietf-oauth-transaction-tokens]
* Workload-Identity-Token

If the response is signed, the following components MUST be signed:

* @status
* @method;req
* @request-target;req

* Content-Type if it exists

* Content-Digest if it exists

* Workload-Identity-Token

To ensure the message is fully integrity-protected, if the request or response includes a message body, the sender MUST include (and the receiver MUST verify) a Content-Digest header.

For both requests and responses, the following signature parameters MUST be included:

* created

* expires - expiration MUST be short, e.g. on the order of minutes. The WIMSE architecture will provide separate mechanisms in support of long-lived compute processes.

* nonce

* tag - the value for implementations of this specification is wimse-workload-to-workload

The following signature parameters in the Signature-Input header MUST NOT be used:

* keyid - The signing key is sent along with the message in the WIT. Additionally specifying the key identity would add confusion.

* alg - The signature algorithm is specified in the jwk section of the cnf claim in the WIT as defined in Section 3.1 "The Workload Identity Token" of [I-D.ietf-wimse-s2s-protocol]. See Sec. 3.3.7 of [RFC9421] for details.

It is RECOMMENDED to include only one signature with the HTTP message. If multiple ones are included, then the signature label included in both the Signature-Input and Signature headers SHOULD be wimse.

A sender MUST ensure that each nonce it generates is unique, at least among messages sent to the same recipient. To detect message replays, a recipient SHOULD reject a message (request or response) if a nonce generated by a certain peer is seen more than once.

Implementors need to be aware that the WIT is extracted from the message before the message signature is validated. Recipients of signed HTTP messages MUST validate the signature and content of the

WIT before validating the HTTP message signature. They MUST ensure that the message is not processed further before it has been fully validated.

Either client or server MAY send an Accept-Signature header, but is not required to do so. When this header is sent, it MUST include the header components listed above.

Following is a non-normative example of a signed request and a signed response, where the caller is using the keys specified in Section 3.1 "The Workload Identity Token" of [I-D.ietf-wimse-s2s-protocol].

```
================ NOTE: '\' line wrapping per RFC 8792 ================

GET /gimme-ice-cream?flavor=vanilla HTTP/1.1
Host: example.com
Signature: wimse=:zyR2W7QeUEYwGT8q+jU5HyEiPhC4iiBJlAfwu6otL+kXBYOMVl\
ZEfGDnTlQiYmUWQMyu69811lIDWbCK42yWCw==:
Signature-Input: wimse=("@method" "@request-target" "workload-identi\
ty-token");created=1754558248;expires=1754558548;nonce="abcd1111";ta\
g="wimse-workload-to-workload"
Workload-Identity-Token: eyJhbGciOiJFZERTQSIsImtpZCI6Imlzc3Vlci1rZXk\
iLCJ0eXAiOiJ3aW1zZS1pZCtqd3QifQ.eyJjbmYiOnsiandrIjp7ImFsZyI6IkVkRFNB\
IiwiY3J2IjoiRWQyNTUxOSIsImtpZCI6InN2Yy1hLWtleSIsImt0eSI6Ik9LUCIsIngi\
OiJEUEhDOFdKaTZ3WEttN3BpRVdXRDRQdHBiTUdJMUcyOXVTMFl5MnNmQk5zIn19LCJl\
eHAiOjE3NTQ1NTg1NDgsImlhdCI6MTc1NDU1ODI0OCwiaXNzIjoiaHR0cHM6Ly9leGFt\
cGxlLmNvbS9pc3N1ZXIiLCJqdGkiOiJ3aXQtMTc1NDU1ODI0ODQwMjQ5NjAwMCIsInN1\
YiI6IndpbXNlOi8vZXhhbXBsZS5jb20vc3ZjQSJ9.OAPARpluzH34v4bhCtPovuOvTsx\
cVbAV9WSqFlbiKdNvfEVN4uRNQSRs7ePfMIJB2uqgX71XjBFH433nDzJZBQ
```

Figure 1: Signed Request

Assuming that the workload being called has the following keypair:

```
{
  "alg": "EdDSA",
  "crv": "Ed25519",
  "d": "JlNJxsZl_PC00EkoRUQbtCrzDtZ5vhFN_6qWtwghttY",
  "kid": "svc-b-key",
  "kty": "OKP",
  "x": "gz2aSJE-g9w1rbgJiNps4Gb8IPk50k5oJUEbLDusayc"
}
```

Figure 2: Callee Private Key

A signed response would be:

```
================ NOTE: '\' line wrapping per RFC 8792 ================

Response:

HTTP/1.1 404 Not Found
Connection: close
Content-Digest: sha-256=:47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU\
=:
Content-Type: text/plain
Signature: wimse=:WAjxziuCiYRqCzetetDwaTS7Ka9yMwB+dAHVJPw3VkUH+c8c4A\
5BKrCsPlD/ymy+7PgwXl3y3mVdaD4ww7WqDA==:
Signature-Input: wimse=("@status" "workload-identity-token" "content\
-type" "content-digest" "@method";req "@request-target";req);created\
=1754558248;expires=1754558550;nonce="abcd2222";tag="wimse-workload-\
to-workload"
Workload-Identity-Token: eyJhbGciOiJFZERTQSIsImtpZCI6Imlzc3Vlci1rZXk\
iLCJ0eXAiOiJ3aW1zZS1pZCtqd3QifQ.eyJjbmYiOnsiandrIjp7ImFsZyI6IkVkRFNB\
IiwiY3J2IjoiRWQyNTUxOSIsImtpZCI6InN2Yy1iLWtleSIsImt0eSI6Ik9LUCIsIngi\
OiJnejJhU0pFLWc5dzFyYmdKaU5wczRHYjhJUGs1MGs1b0pVRWJMRHVzYXljIn19LCJl\
eHAiOjE3NTQ1NTg1NTAsImlhdCI6MTc1NDU1ODI1MCwiaXNzIjoiaHR0cHM6Ly9leGFt\
cGxlLmNvbS9pc3N1ZXIiLCJqdGkiOiJ3aXQtMTc1NDU1ODI0ODQwMjY4MTAwMCIsInN1\
YiI6IndpbXNlOi8vZXhhbXBsZS5jb20vc3ZjQiJ9.yJ4gGQ1DQBX0E9Xp5fIgj6Ucpf9\
LI_gHnhURZxqu6atT-3hpbFTgw4rd-6knM7-HClok4b6N2ViZaEDcz6IMCg

No ice cream today.
```

Figure 3: Signed Response

3.1. Error Conditions

Errors may occur during the processing of the message signature or WPT. If the signature verification fails for any reason, such as an invalid signature, an expired validity time window, or a malformed data structure, an error is returned. Typically, this will be in response to an API call, so an HTTP status code such as 400 (Bad Request) is appropriate. This response could include more details as per [RFC9457], such as an indicator that the wrong key material or algorithm was used. The use of HTTP status code 401 is NOT RECOMMENDED for this purpose because it requires a WWW-Authenticate with acceptable http auth mechanisms in the error response and an associated Authorization header in the subsequent request. The use of these headers for the WIT or WPT is not compatible with this specification.

3.2. Coexistence with JWT Bearer Tokens

The WIT and WPT define new HTTP headers. They can therefore be presented along with existing headers used for JWT bearer tokens. This property allows for transition from mechanisms using identity tokens based on bearer JWTs to proof of possession based WITs. A workload may implement a policy that accepts both bearer tokens and WITs during a transition period. This policy may be configurable per-caller to allow the workload to reject bearer tokens from callers that support WITs. Once a deployment fully supports WITs, then the use of bearer tokens for identity can be disabled through policy. Implementations should be careful when implementing such a transition strategy, since the decision which token to prefer is made when the caller's identity has still not been authenticated, and needs to be revalidated following the authentication step.

The WIT can also coexist with tokens used to establish security context, such as transaction tokens [I-D.ietf-oauth-transaction-tokens]. In this case a workload's authorization policy may take into account both the sending workload's identity and the information in the context token. For example, the identity in the WIT may be used to establish which API calls can be made and information in the context token may be used to determine which specific resources can be accessed.

4. Security Considerations

4.1. Workload Identity Token and Proof of Possession

The Workload Identity Token (WIT) is bound to a secret cryptographic key and is always presented with a proof of possession as described in Section 3.1 "The Workload Identity Token" of [I-D.ietf-wimse-s2s-protocol]. The WIT is a general purpose token that can be presented in multiple contexts. The WIT and its PoP are only used in the application-level options, and both are not used in MTLS. The WIT MUST NOT be used as a bearer token. While this helps reduce the sensitivity of the token it is still possible that a token and its proof of possession may be captured and replayed within the PoP's lifetime. The following are some mitigations for the capture and reuse of the proof of possession (PoP):

* Preventing Eavesdropping and Interception with TLS

An attacker observing or intercepting the communication channel can view the token and its proof of possession and attempt to replay it to gain an advantage. In order to prevent this the token and proof of possession MUST be sent over a secure, server authenticated TLS connection unless a secure channel is provided by some other

mechanisms. Host name validation according to Section 3.3.1 "Server Name Validation" of [I-D.ietf-wimse-s2s-protocol] MUST be performed by the client.

* Limiting Proof of Possession Lifespan

The proof of possession MUST be time limited. A PoP should only be valid over the time necessary for it to be successfully used for the purpose it is needed. This will typically be on the order of minutes. PoPs received outside their validity time MUST be rejected.

* Limiting Proof of Possession Scope

In order to reduce the risk of theft and replay the PoP should have a limited scope. For example, a PoP may be targeted for use with a specific workload and even a specific transaction to reduce the impact of a stolen PoP. In some cases a workload may wish to reuse a PoP for a period of time or have it accepted by multiple target workloads. A careful analysis is warranted to understand the impacts to the system if a PoP is disclosed allowing it to be presented by an attacker along with a captured WIT.

* Replay Protection

A proof of possession includes the jti claim that MUST uniquely identify it, within the scope of a particular sender. This claim SHOULD be used by the receiver to perform basic replay protection against tokens it has already seen. Depending upon the design of the system it may be difficult to synchronize the replay cache across all token validators. If an attacker can somehow influence the identity of the validator (e.g. which cluster member receives the message) then replay protection would not be effective.

* Binding to TLS Endpoint

The POP MAY be bound to a transport layer sender such as the client identity of a TLS session or TLS channel binding parameters. The mechanisms for binding are outside the scope of this specification.

4.2. Middle Boxes

In some deployments the Workload Identity Token and proof of possession may pass through multiple systems. The communication between the systems is over TLS, but the token and PoP are available in the clear at each intermediary. While the intermediary cannot modify the token or the information within the PoP they can attempt to capture and replay the token or modify the data not protected by the PoP.

Mitigations listed in Section 4.3 "Middle Boxes" of [I-D.ietf-wimse-s2s-protocol] can be used to provide some protection from middle boxes. However we note that the DPoP-inspired solution [I-D.ietf-schwenkschuster-s2s-jwt-pop] does not protect major portions of the request and response and therefore does not provide protection from an actively malicious middle box. Deployments should perform analysis on their situation to determine if it is appropriate to trust and allow traffic to pass through a middle box.

4.3. Privacy Considerations

Privacy considerations for this specification are the same as those described in Section 4.4 "Privacy Considerations" of [I-D.ietf-wimse-s2s-protocol].

5. IANA Considerations

None

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/rfc/rfc7515>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <https://www.rfc-editor.org/rfc/rfc7517>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <https://www.rfc-editor.org/rfc/rfc7518>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

[RFC7800] Jones, M., Bradley, J., and H. Tschofenig, "Proof-of-Possession Key Semantics for JSON Web Tokens (JWTs)", RFC 7800, DOI 10.17487/RFC7800, April 2016, <https://www.rfc-editor.org/rfc/rfc7800>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/rfc/rfc8725>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[RFC9421] Backman, A., Ed., Richer, J., Ed., and M. Sporny, "HTTP Message Signatures", RFC 9421, DOI 10.17487/RFC9421, February 2024, <https://www.rfc-editor.org/rfc/rfc9421>.

6.2. Informative References

[I-D.ietf-oauth-transaction-tokens] Tulshibagwale, A., Fletcher, G., and P. Kasselman, "Transaction Tokens", Work in Progress, Internet-Draft, draft-ietf-oauth-transaction-tokens-06, 28 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-transaction-tokens-06>.

[I-D.ietf-schwenkschuster-s2s-jwt-pop] "*** BROKEN REFERENCE ***".

[I-D.ietf-wimse-arch] Salowey, J. A., Rosomakho, Y., and H. Tschofenig, "Workload Identity in a Multi System Environment (WIMSE) Architecture", Work in Progress, Internet-Draft, draft-ietf-wimse-arch-05, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-arch-05>.

[I-D.ietf-wimse-s2s-protocol]
Campbell, B., Salowey, J. A., Schwenkschuster, A., and Y. Sheffer, "WIMSE Workload to Workload Authentication", Work in Progress, Internet-Draft, draft-ietf-wimse-s2s-protocol-06, 4 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-s2s-protocol-06>.

[IANA.HTTP.FIELDS]
IANA, "Hypertext Transfer Protocol (HTTP) Field Name Registry", <https://www.iana.org/assignments/http-fields>.

[IANA.JOSE.ALGS]
IANA, "JSON Web Signature and Encryption Algorithms", <https://www.iana.org/assignments/jose>.

[IANA.JWT.CLAIMS]
IANA, "JSON Web Token Claims", <https://www.iana.org/assignments/jwt>.

[IANA.MEDIA.TYPES]
IANA, "Media Types", <https://www.iana.org/assignments/media-types>.

[IANA.URI.SCHEMES]
IANA, "Uniform Resource Identifier (URI) Schemes", <https://www.iana.org/assignments/uri-schemes>.

[RFC9457] Nottingham, M., Wilde, E., and S. Dalal, "Problem Details for HTTP APIs", RFC 9457, DOI 10.17487/RFC9457, July 2023, <https://www.rfc-editor.org/rfc/rfc9457>.

Appendix A. Document History

// RFC Editor: please remove before publication.

A.1. draft-schwenkschuster-s2s-http-sig-00

Initial clone from original draft-ietf-wimse-s2s-06 which contained both, Workload Proof Token and HTTP Message Signature proof of possession mechanisms.

Acknowledgments

The authors would like to thank Pieter Kasselman for his detailed comments.

We thank Daniel Feldman for his contributions to earlier versions of this document.

Author’s Address

Arndt Schwenkschuster
SPIRL
Email: arndts.ietf@gmail.com